Towards high throughput semantic translation

Maria Ganzha[1,4], Marcin Paprzycki[1,5], Wiesław Pawłowski[2], Paweł Szmeja[1], Katarzyna Wasielewska[1], Bartłomiej Solarz-Niesłuchowski[1], and Jara Suárez de Puga García[3]

[1] Systems Research Institute, Polish Academy of Sciences
firstname.lastname@ibspan.waw.pl
[2] Faculty of Mathematics, Physics, and Informatics, University of Gdańsk
wieslaw.pawlowski@inf.ug.edu.pl
[3] Departamento de Comunicaciones, Universitat Politècnica de València
jasuade@dcom.upv.es
[4] Warsaw University of Technology
[5] Warsaw Management Academy

Abstract. One of "urban legends" of today's computer science is: *use of semantic technologies can become a serious performance bottleneck*. It is even possible that this, widely spread, belief is one of the reasons of slow progress in adopting semantic technologies in real-world applications. Since, obviously, IoT scenarios involve fast flowing streams of sensor data, will use of semantic technologies be "efficient enough" to not to adversely affect the effectiveness of the whole Internet of Things (IoT) ecosystem. The aim of this contribution is to provide an initial response to this question.

1 Introduction

It is rather rare when semantic technologies materialize in real-world/industrial grade projects/deployments. Instead, they are one of poster children of academic research. It is easy to find publications claiming that, for instance, ontologies will become a silver bullet for problems of e-commerce [3], or that semantics combined with software agents will deliver novel form of intelligent systems [2]. It can be stipulated that one of the reasons of this situation is a, widely held, belief that semantic technologies result in poor performance of applications. However, very few tests of this claim, for specific deployments, have been run.

In this context, in the (EU-funded) INTER-IoT project it was decided that semantic technologies will facilitate high-level interoperability between IoT artifacts (platforms, middlewares, applications, etc.). As reported in [4,6,5,10] the proposed approach can be summarized as follows. Assume that (1) multiple artifacts have to communicate (exchange/send/receive) messages within an IoT ecosystem; and (2) majority of them use different internal data representation (syntax and semantics). Then, to facilitate interoperability, the following steps need to be undertaken. (1) Semantics of each artifact has to be represented in the form of an OWL-based ontology. This may require "lifting" other data representations to OWL ontologies [4,7]. (2) Uni- or bi-directional syntactic translators

have to be implemented, to translate data from the artifact's internal syntax to the RDF representation (and, possibly, back). Note that when an artifact already uses RDF, the process is reduced to wrapping the message into a "proper structure". Need for uni- or bi-directional translation depends on the flow of information to be used. (3) Central modular ontology, covering the "core concepts of the IoT" as well as "domain specific" aspects of the deployment, has to be designed (from modules recommended by INTER-IoT, or other external ontologies). Modularity of the central (OWL-based) ontology assures that individual message flows can be easier managed, if they concern different "aspects" of the ecosystem. (4) Semantic translation mechanism, based on ontology alignments [5,10], has to be facilitated for translations between ontologies representing semantics of individual artifacts and the central ontology (and back). Here, again, need for uni- or bi-directional translation depends on the flow of messages between artifacts. The latter functionality has been conceptualized and implemented in the Inter-Platform Semantic Mediator (IPSM) component. The translation rules are defined in an alignment persisted in the INTER-IoT Alignment Format.

The aim of this contribution is to described preliminary experiments measuring performance of the IPSM. In the next section we introduce the architecture of the IPSM. We follow with results of throughput testing and conclude with proposed future work.

2 Inter-Platform Semantic Mediator

The role of the Inter-Platform Semantic Mediator is to facilitate alignment-based translation between source and target artifacts' semantics. Specifically, for each "communicating artifact" it's ontology is aligned with the central ontology. Note that there is no need to define complete alignments between ontologies. It is enough to capture correspondences between parts (or modules) used in communication. Specifically, while an alignment can represent number of correspondences (for a pair of ontologies), only the actually used ones are included. Alignments are represented in the INTER-IoT Alignment Format (for details see [5,10]). This format is based on the Alignment API[6] and influenced by the EDOAL[7].

The architecture of the IPSM is depicted in Figure 1, where one can "follow" the translation process. Briefly, IPSM offers *translation channels* and communication infrastructure, based on Apache Kafka[8]. An input RDF message, expressed in the source ontology, is published to a preconfigured *input topic* associated with a translation channel. Next, it gets translated to an RDF message, in the target ontology, and published to a preconfigured *output topic* of the channel, from where it can be consumed. Operations, for alignments and channel management, are exposed via a REST API. Each communication channel has a number of configuration parameters, including input and output topic names, alignment from source to central ontology, alignment from central to target ontology, and

[6] http://alignapi.gforge.inria.fr/format.html
[7] http://alignapi.gforge.inria.fr/edoal.html
[8] https://kafka.apache.org/

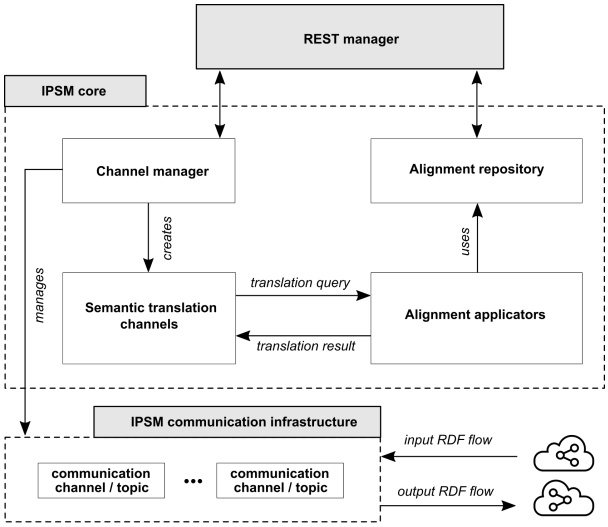

Fig. 1. IPSM architecture

a "parallelism factor" of the channel. A more detailed description of the IPSM inner structure and the translation process can be found in [6].

3 IPSM throughput testing

Let us now discuss results of experimental testing of performance of the IPSM. Since IPSM is used to translate messages, we have decided to focus on throughput of the translation process. In other words, we assess how fast messages are translated. Due to the limited space, we report only a single set of experiments.

3.1 Experimental setup

Let us introduce the setup used for carrying the experiments. We have used three different "machines". (A) Desktop PC with dual-core AMD Athlon 64 X2 processor and 4GB of RAM, (B) Desktop PC with a quad-core Intel Core2 processor running at 2.4GHz, with 4GB of RAM, and (C) MS Azure VM exposing a "dual-core subset" of the Intel Xeon E5-2673 processor[9], and 8GB of RAM.

Translation involved real-life size and complexity messages, including geospatial data. In each experiment, 40,000 messages have been generated (with actual payload, differing by randomly chosen sensor data and numerical values of latitude and longitude). IPSM was set up to log the individual message processing time using Apache Kafka. Monitoring data was analyzed using R[10] framework.

[9] 12-core Xeon E5-2673 offers two logical cores for each physical one, resulting in 4 (logical) cores available to the Azure VM.

[10] https://www.r-project.org/

3.2 Results and analysis

In Figure 2, a comparative histogram of message processing times for all three testing environments is presented. Specifically, we show how many messages have been processed within 5 millisecond, 15 milliseconds, 25 milliseconds, etc. As expected, the Azure VM was the most efficient, with a majority of messages

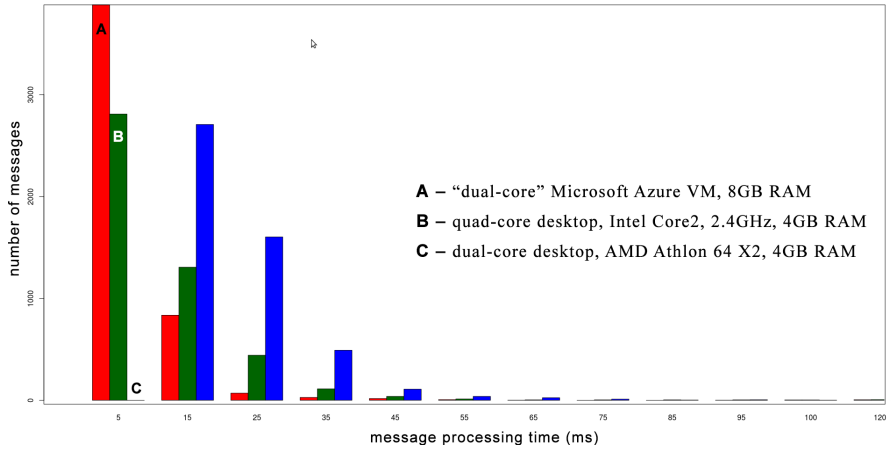

Fig. 2. IPSM message processing time(s) – comparative histogram

processed within 5-15 milliseconds. The next was the quad-core Intel machine, where processing time of majority of messages was between 5-25 milliseconds. More importantly, even in the case of a weak, approximately 8 years old Athlon-based machine, the processing time of a vast majority of messages was less than 35 milliseconds. This seems to contradict that general view that semantic technologies have to introduce bottlenecks to applications running them.

Interestingly, we have observed (but have no space to depict) that performance of the Azure-based infrastructure was somewhat "erratic". Specifically, in each run there was a (small) number of individual messages that took relatively large amount of time to be translated. For instance, 3-5 messages took around 200 milliseconds each, to be processed. We believe that this is an effect of Azure being a virtual machine, running together with other processes/VMs on a shared hardware, and competing for resources. No similar "obvious processing time peaks" were observed for the "physical machines". The figure shows that the IPSM performance "gently degrades" in accordance with the hardware specification of the machines used for testing.

The next series of experiments concerned possibility of further utilizing potential of the underlying hardware. We started by testing the influence of *internal parallelism* of translation channels. This time, we have concentrated solely on the

Azure VM, as the most powerful environment in our setup. Results of experiments have been summarized in Figure 3. Once again, we have streamed 40 thousand randomly generated, parametrized messages through a single translation channel, with a "parallelism factor" ranging from 1, up to 32. Figure 3 shows, for each of the tested factor values, a relationship between the message processing time *median*, and the *total time* needed to translate the whole stream.

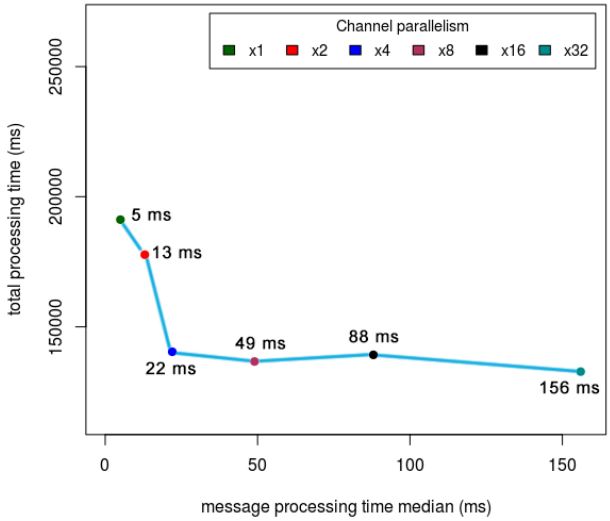

Fig. 3. Processing 40k messages via channel with internal parallelism on Azure VM

As can be seen, "performance gain" (represented as reduction of total processing time for all messages) has a "sweet spot" for the channel parallelism factor equal to 4. Further attempts at increasing the channel internal parallelism either result in substantial increase of the median or do not bring (sufficient) reduction of the total processing time. In other words, in-channel 4x parallelism exhausted resources of hardware used in our experiments. Obviously, for different hardware, it may be possible to further (in-channel) parallelize flow of messages.

Finally, in the last series of experiments, we have used multiple 1-16 (purely sequential) IPSM channels. As before, each run involved 40.000 messages.

The results show that the sweet spot is, again, reached for four channels, where the underlying hardware became "saturated". Together with the results of the channel internal parallelism tests this shows that the IPSM translation infrastructure is able to efficiently utilize the "native threads" (logical cores) of the VM's CPU.

All completed, thus far, performance test show that the IPSM architecture is not only efficient, but also scales very well "vertically", i.e., the more powerful the

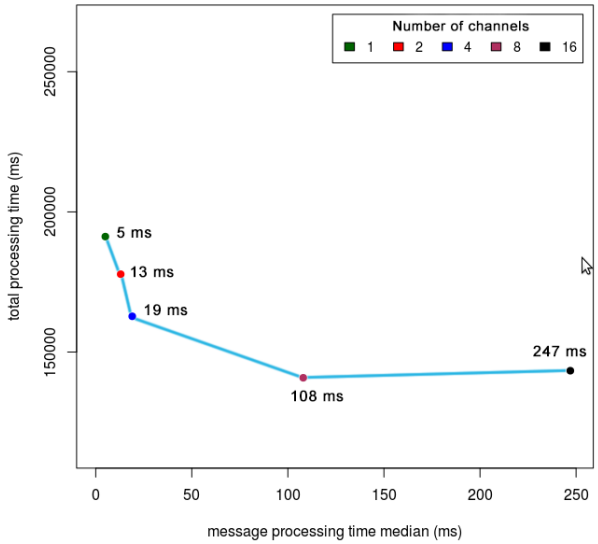

Fig. 4. Processing 40k messages via multiple channels on Azure VM

available hardware, the better translation throughput can be observed. Here, let us note that IPSM has been implemented using Scala programming language [9] and the actor-based [8,1] Akka[11] toolkit. Thanks to the well known properties of Akka and Apache Kafka, which was used to organize the IPSM communication infrastructure, and due to the fact that IPSM is component that can be deployed in multiple instances, it should also be highly scalable "horizontally".

4 Conclusions and future work

In this contribution we have attempted at experimentally addressing the question: is it really the case that use of semantic technologies results in serious performance bottlenecks? The context for seeking an answer was provided by the semantic translation that is being implemented in the IPSM component, developed within the framework of the INTER-IoT project.

The results are quite promising, since even when running the IPSM on an almost obsolete computer hardware, the processing time is reasonable. Therefore, we believe that semantic technologies can be used for information processing (e.g. semantic translation) even in the case relatively fast data flows.

Obviously, it is possible that more complex semantic translation scenarios than the ones used in our tests, will be needed. As a matter of fact, such translations are very likely to materialize within the scope of the INTER-IoT project. Therefore, we will continue our research and study performance of the IPSM in

[11] https:/akka.io/

such cases. We will also see, which elements of the IPSM can be optimized to further improve its performance (in case of complex semantic translations). To even further improve the horizontal scalability of our solution we plan to investigate the possibility of forming IPSM clusters, where multiple IPSM instances can be administered and utilized in a uniform way.

Acknowledgment

This research was partially supported by the European Union's "Horizon 2020" research and innovation program as part of the "Interoperability of Heterogeneous IoT Platforms" (INTER-IoT) project under Grant Agreement No. 687283.

References

1. Agha, G.A.: Actors: A Model of Concurrent Computation in Distributed Systems. MIT Press, Cambridge, MA, USA (1986)
2. Allemang, D., Handler, J.: Semantic Web for the Working Ontologist, Second Edition: Effective Modeling in RDFS and OWL. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2008)
3. Fensel, D.: Ontologies: Silver Bullet for Knowledge Management and Electronic Commerce. Springer-Verlag, Berlin (2000)
4. Ganzha, M., Paprzycki, M., Pawłowski, W., Szmeja, P., Wasielewska, K.: Towards semantic interoperability between Internet of Things platforms. In: Gravina, R., Palau, C.E., Manso, M., Liotta, A., Fortino, G. (eds.) Integration, Interconnection, and Interoperability of IoT Systems, pp. 103–127. Springer (2017)
5. Ganzha, M., Paprzycki, M., Pawłowski, W., Szmeja, P., Wasielewska, K.: Alignment-based semantic translation of geospatial data. In: 3rd International Conference on Advances in Computing, Communication & Automation (ICACCA), Proceedings (in press)
6. Ganzha, M., Paprzycki, M., Pawłowski, W., Szmeja, P., Wasielewska, K.: Streaming semantic translations. In: 21st International Conference on System Theory, Control and Computing ICSTCC, Proceedings (in press)
7. Ganzha, M., Paprzycki, M., Pawłowski, W., Szmeja, P., Wasielewska, K., Palau, C.E.: From implicit semantics towards ontologies—practical considerations from the INTER-IoT perspective (submitted for publication). In: Proc. of 1st edition of Globe-IoT 2017: Towards Global Interoperability among IoT Systems (2017)
8. Hewitt, C., Bishop, P., Steiger, R.: A universal modular ACTOR formalism for artificial intelligence. In: Proceedings of the 3rd International Joint Conference on Artificial Intelligence. pp. 235–245. IJCAI'73, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1973)
9. Odersky, M., Spoon, L., Venners, B.: Programming in Scala. Artima Press, USA, 3rd edn. (2016)
10. Szmeja, P., Ganzha, M., Paprzycki, M., Pawłowski, W., Wasielewska, K.: Declarative ontology alignment format for semantic translation. In: 6th EAI International Conference on Context-Aware Systems and Applications (ICCASA), Proceedings (in press)